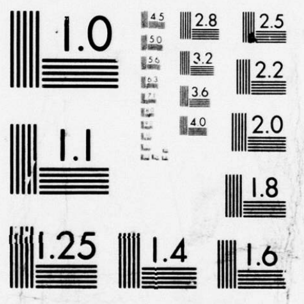
AD-A060 148 GEORGE WASHINGTON UNIV WASHINGTON D C INST FOR MANAG--ETC F/G 12/1
LARGE SCALE NONLINEAR PROGRAMMING.(U)
JUN 78 G P MCCORMICK DAAG29-76-G-0150

UNCLASSIFIED SERIAL-T-375 ARO-13622.5-M NL

1 OF 1
AD
A060148

END
DATE
FILMED
12-78
DDC

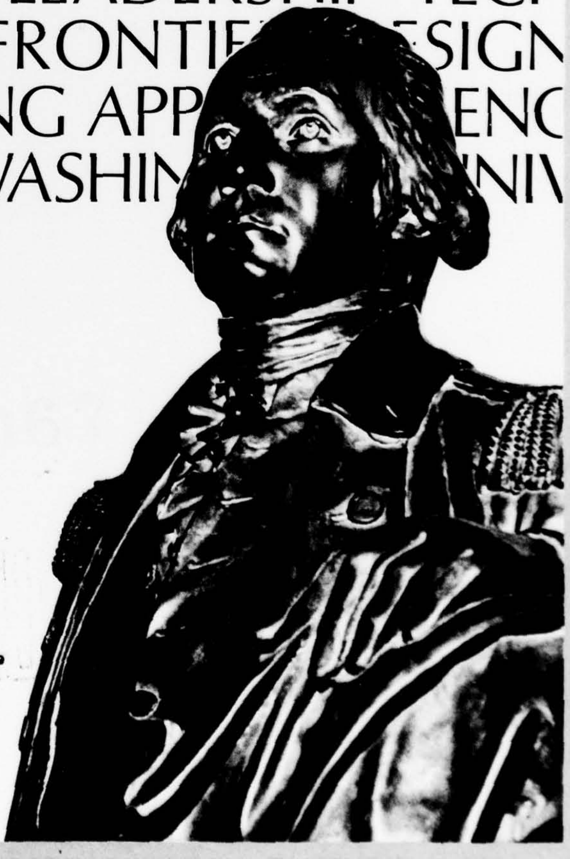

1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

(12) ARO 13622.3-M (11)

THE GEORGE WASHINGTON UNIVERSITY

LEVEL II

STUDENTS FACULTY STUDY R
ESEARCH DEVELOPMENT FUT
URE CAREER CREATIVITY CO
MMUNITY LEADERSHIP TECH
NOLOGY FRONTI ESIGN
ENGINEERING APP ENC
GEORGE WASHI NIV



INSTITUTE FOR MANAGEMENT SCIENCE AND ENGINEERING
SCHOOL OF ENGINEERING AND APPLIED SCIENCE

LEVEL II

6 LARGE SCALE NONLINEAR PROGRAMMING

by

10 Garth P./McCormick

14 SERIAL-T-375

~~Serial~~ T-378

11 15 Jun[redacted]78

12 18p.

9 Scientific rept.,

18 ARO

19 13622.5-M

The George Washington University
School of Engineering and Applied Science
Institute for Management Science and Engineering

Research Sponsored by
U.S. Army Research Office
Contract 15 DAAG29-76-G-0150

78 10 10 067

406 743 [illegible]

NONE

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ~~T-378~~ / 2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (*and Subtitle*) LARGE SCALE NONLINEAR PROGRAMMING	5. TYPE OF REPORT & PERIOD COVERED SCIENTIFIC
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) GARTH P. McCORMICK	8. CONTRACT OR GRANT NUMBER(s) DAAG29-76-G-0150
9. PERFORMING ORGANIZATION NAME AND ADDRESS THE GEORGE WASHINGTON UNIVERSITY INSTITUTE FOR MANAGEMENT SCIENCE & ENGINEERING WASHINGTON, D. C. 20052	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. ARMY RESEARCH OFFICE BOX 12211 RESEARCH TRIANGLE PARK, N. C. 27709	12. REPORT DATE 15 JUNE 1978
	13. NUMBER OF PAGES 15
14. MONITORING AGENCY NAME & ADDRESS(*if different from Controlling Office*)	15. SECURITY CLASS. (*of this report*) NONE
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (*of this Report*)

DISTRIBUTION OF THIS REPORT IS UNLIMITED.

17. DISTRIBUTION STATEMENT (*of the abstract entered in Block 20, if different from Report*)

ACCESSION for
NTIS White Section
DDC Buff Section
UNANNOUNCED
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
Dist. AVAIL. and/or SPECIAL
A

18. SUPPLEMENTARY NOTES

19. KEY WORDS (*Continue on reverse side if necessary and identify by block number*)

LARGE SCALE OPTIMIZATION
NONLINEAR PROGRAMMING
APPLICATIONS OF NONLINEAR PROGRAMMING

20. ABSTRACT (*Continue on reverse side if necessary and identify by block number*)

This paper considers first the real world situations which give rise to large nonlinear programming problems. Next the algebraic structure of these problems is analyzed. A brief survey of algorithms for solving them is given with emphasis on those whose computer programs are available. Concluding remarks concern an estimation of future research required in this area.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

NONE

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

THE GEORGE WASHINGTON UNIVERSITY
School of Engineering and Applied Science
Institute for Management Science and Engineering

Abstract
of
Serial T-378
15 June 1978

LARGE SCALE NONLINEAR PROGRAMMING

by

Garth P. McCormick

This paper considers first the real world situations which give rise to large nonlinear programming problems. Next the algebraic structure of these problems is analyzed. A brief survey of algorithms for solving them is given with emphasis on those whose computer programs are available. Concluding remarks concern an estimation of future research required in this area.

Research Sponsored by
U.S. Army Research Office
Research Triangle Park, North Carolina 27709

THE GEORGE WASHINGTON UNIVERSITY
School of Engineering and Applied Science
Institute for Management Science and Engineering

LARGE SCALE NONLINEAR PROGRAMMING

by

Garth P. McCormick

1. Introduction

The general mathematical programming (optimization) problem can be stated in the following form: find values $(\bar{x}_1,\ldots,\bar{x}_n)$ which solve

$$\underset{x}{\text{minimize}}\quad f(x)$$

subject to the restrictions that (1)

$$g_i(x) \geqq 0\ , \quad \text{for}\quad i=1,\ldots,m\ ,$$

and

$$h_j(x) = 0\ , \quad \text{for}\quad j=1,\ldots,p\ .$$

What is meant by a "large" optimization problem depends upon the nature of the functions f , $\{g_i\}$, and $\{h_j\}$. If they are linear functions, it is usually the case that the restrictions $x_j \geqq 0$ for $j=1,\ldots,n$ are included in the inequality constraints and the problem is considered large when $(m+p-n)$ is more than 2,000. For linear programming problems, the size of n is, for all practical purposes, unlimited. When the functions are nonlinear, the size of n becomes important, and a large problem can be one where n is above 50. An exact definition is hard to give because the difficulty in solving a general nonlinear optimization problem has as much to do with the nature of the functions involved as it does with the dimensions of the problems. It is best first to describe situations which give rise to large nonlinear programming problems and analyze their algebraic structure. This is done in Section 2. In Section 3 there is an overview of current computer coded algorithms for solving large nonlinear programs. In Section 4 is some speculation on future research needed in this area with emphasis on a new approach called factorable programming.

2. Large Nonlinear Programming Models

A major source of large nonlinear (and linear also) optimization problems comes from the general area known as "resource allocation." The variables of the problem are usually written as $\{x_{k\ell}\}$, where $k=1,\ldots,p$, and $\ell=1,\ldots,q$. Here p is the number of resources available, and q is the number of tasks or missions which can utilize the resources. The functions f , $\{g_i\}$, $\{h_j\}$ are usually cost functions, effectiveness functions, and functions which represent constraints on the resources. The number of nontrivial constraints is usually of the order of $(p+q)$, but the problem can become "large" quickly because the number of variables is equal to $p \cdot q$. The following [Bracken and McCormick 1968] is a simple example in the area of weapons allocation which illustrates this class of problems.

Variables.

$x_{k\ell}$ = number of weapons of type k to be allocated to target ℓ , for $k=1,\ldots,p;\ \ell=1,\ldots,q$.

Fixed inputs.

a_k = (for $k=1,\ldots,p$) total weapons of type k available

$\alpha_{k\ell}$ = the probability that target ℓ will be undamaged by an attack of one of weapon type k

u_ℓ = the military value of target ℓ .

The optimization problem is then

$$\underset{\{x_{k\ell}\}}{\text{maximize}} \quad \textstyle\sum_{\ell=1}^{q} u_\ell \left\{1 - \prod_{k=1}^{p} \alpha_{k\ell}^{x_{k\ell}}\right\} \quad \text{(expected target damage value)}$$

$$\text{subject to} \quad \textstyle\sum_{\ell=1}^{q} x_{k\ell} \leqq a_k \;, \quad \text{for } k=1,\ldots,p$$

$$\text{and} \quad x_{k\ell} \geqq 0 \qquad , \quad \text{for } k=1,\ldots,p;\ \ell=1,\ldots,q \;.$$

Problems of this type typically have a special structure. In this case the constraints are linear and "sparse." The density of the matrix of nontrivial linear constraints is $1/p$ with only the number "1." defining the matrix. The constraints are all of the form of Generalized Upper Bounds (GUB). Algorithms for solving linear problems have been able recently to take advantage of this.

The concept of sparseness for nonlinearly constrained problems has not been developed in general. This example will be examined further when it is shown how to use linear programming to solve separable optimization problems.

Another important area of nonlinear programming is that of optimal structural design. This is an area where the ability to generate large difficult problems has outstripped the ability of algorithmists to solve them. A typical problem takes the following form. Let α be a vector of design variables, and x a vector of state variables. The usual objective function in design problems is weight (to be minimized). The design variables are often cross-sectional areas of the members of the structure. The constraints usually involve compatibility, equilibrium, and force deformation.

Variables.

α = design variables

x = state variables (in one formulation these are eliminated from the problem)

Fixed inputs and derived quantities.

$S(\alpha)$	=	a positive definite matrix which depends on the design variables
A	=	a matrix which depends upon the geometry of the problem
M	=	mass matrix
P	=	vector of applied forces
σ	=	allowable stress for material used
Δ	=	vector of allowable deflections for state variables
$K(\alpha)$	=	$A^T S(\alpha) A$, the stiffness matrix
L_i	=	length of ith member
ρ	=	density of material.

The optimization problem is then

$$\begin{aligned} \underset{(\alpha,x)}{\text{minimize}} \quad & \rho \sum_i L_i \alpha_i && \text{(Weight)} \\ \text{subject to} \quad & x \leqq \Delta && \text{(Deflection Constraints)} \\ & K(\alpha)x = P && \text{(Stiffness Constraints)} \\ & S(\alpha)Ax \leqq \alpha\sigma && \text{(Stress Constraints)} \end{aligned}$$

Another interesting complication occurs if a buckling constraint is imposed. This takes the form $\lambda_{min}(\alpha) \geqq$ (some specified force) , where $\lambda_{min}(\alpha)$ is the smallest eigenvalue of the generalized eigenvalue problem $K(\alpha)y = My\lambda$.

For large structures, the number of variables and constraints becomes quite large. An alternative formulation which reduces the number of variables is to solve for $x = K(\alpha)^{-1}P$ and substitute this wherever x appears. There are difficulties in obtaining derivatives for this implicitly defined problem, but it can be done with some effort and thought.

Use of finite element techniques for shell structures is a source of large nonlinear optimization problems.

There is an enormous literature on structural design. A sample of material in the area is contained in [Haug 1973] and [Cohn and Maier 1978].

The final example of large nonlinear optimization problems comes from the logistics area, inventory control. In [Schrady and Choe 1971] the following problem was formulated.

Variables.

Q_i, r_i	for $i=1,\ldots,N$, the amount to order and the reorder point of the ith inventory item.

Constants and fixed inputs.

c_i	=	item unit cost
λ_i	=	mean demand per unit time (in units)
K_1	=	maximum amount of money allowed to be tied up in inventory
K_2	=	reorder workload limit allowed to be tied up in inventory
μ_i, σ_i	=	mean and standard deviation of lead time demand (which is assumed normally distributed) for the ith item.

Let $\phi(x)$ denote the normal density function with mean 0, standard deviation 1., and let $\Phi(z) = \int_z^\infty \phi(x)\,dx$. The optimization problem which produces the optimal order quantities and reorder points is

$$\underset{\{Q_i, r_i\}}{\text{minimize}} \quad f = \sum_{i=1}^N \frac{1}{2Q_i} \left[\left\{ \sigma_i^2 + (r_i - \mu_i)^2 \right\} \Phi\left(\frac{r_i - \mu_i}{\sigma_i} \right) - \sigma_i (r_i - \mu_i)\, \phi\left(\frac{r_i - \mu_i}{{}_i} \right) \right]$$

(expected time-weighted shortages)

subject to $g_1 = K_2 - \sum_{i=1}^N \lambda_i / Q_i \geq 0$ (reorder workload constraint)

$g_2 = K_1 - \sum_{i=1}^N c_i (r_i + Q_i/2 - \mu_i) \geq 0$ (investment constraint)

$Q_i \geq 0$, $i=1,\ldots,N$.

When N , the number of inventory items, is large, this generates a large nonlinear programming problem. It has a special structure which can be utilized to solve it. This is discussed later in the paper when the SUMT algorithm is applied to solve the problem.

Any nonlinear programming problem which has a dynamic character, i.e., whose variables are time-dependent, is usually large since the number of variables is equal to the product of the "x's" times the number of time periods. Examples of this are problems in optimal control, the calculus of variations, and economic planning models. More sophisticated dynamic versions of the weapons assignment problem with elements of game theory and max-min also generate large problems. See [Tomlin 1978] for a discussion of this. These problems generally have a decomposable structure which can be used to uncouple the problem. Since one of the papers in this meeting concerns that approach it will not be discussed further here.

3. Computer Coded Algorithms for Solving Large Nonlinear Programming Problems

There are direct and indirect methods for solving large nonlinear optimization problems. The indirect ways consist usually of solving a sequence of linear or linearly constrained problems since computer programs for solving these problems are highly developed. The simplest way of doing this was proposed in [Griffith and Stewart 1961]. The idea is to make local linear approximations to the objective function and constraints and solve the resulting linear programming problems.

The second way is to take a general nonlinear optimization problem and solve it using a sequence of linearly constrained nonlinear programming problems. Recent codes for solving the latter are [Lasdon and Warren 1975] and [Murtagh and Saunders 1978]. Some papers relating to the theory and algorithmic developments for converting nonlinearly constrained problems into sequences of linearly constrained problems are discussed in [Powell 1978].

The third approach is to take a nonlinear programming problem, convert it to an equivalent separable nonlinear programming problem by adding equality constraints and extra variables, then solve it using the separable programming capability which is available with some of the large-scale linear programming systems [Beale 1975]. Just about any nonlinear programming problem can be "separated" using some simple tricks. To illustrate this, consider the weapons assignment problem of Section 1.

Use is made of the identity, $\alpha^x = e^{x \ell n \alpha}$. First, convert the data and let $\beta_{k\ell} = \ell n\ \alpha_{k\ell}$, for $k=1,\ldots,p$, and $\ell=1,\ldots,q$. Introduce the new variables $\{z_\ell\}$, and form the equality constraints $z_\ell = \sum_{k=1}^{p} x_{k\ell}\beta_{k\ell}$, for $\ell=1,\ldots,q$. The equivalent separable optimization problem now has the form

$$\underset{\{x_{k\ell}\},\{z_\ell\}}{\text{maximize}} \quad \sum_{\ell=1}^{q} u_\ell [1 - e^{z_\ell}]$$

subject to the restrictions that

$$\sum_{\ell=1}^{q} x_{k\ell} \leqq a_k \text{ , for } k=1,\ldots,q \text{ ,} \qquad z_\ell = \sum_{k=1}^{p} x_{k\ell}\beta_{k\ell} \text{ , for } \ell=1,\ldots,q \text{ ,}$$

and

$$x_{k\ell} \geqq 0 \text{ , for } k=1,\ldots,p;\ \ell=1,\ldots,q \text{ .}$$

The new problem has $pq + q$ variables, and $p + q$ nontrivial linear constraints.

Depending upon the problem, the creation of an equivalent separable one may generate a "large" separable problem from a moderate sized nonseparable one. For discussion of a general systematic approach for separating nonlinear programming problems, see [McCormick 1972b].

Separable problems are solved by the large linear programming systems using the device of approximating the nonlinear functions of a single variable by piecewise linear functions. There are many approaches for doing this. References and discussion of this are in [Beale 1975].

A direct approach for solving large nonlinear programming problems generally consists of taking advantage of the special structure of a particular large problem and modifying the linear algebra (matrix techniques) used for solving the problem so that the number of operations (multiplications and additions) is reduced, as well as obtaining a reduction in storage. To illustrate this, consider the penalty function approach to the inventory problem of Section 2. Let x be the vector of variables $\{Q_i, r_i\}$.

The penalty function approach is to find the unconstrained minimizer of

$$P_k = f - r_k \ell n(g_1) - r_k \ell n(g_2) - \sum_{i=1}^{N} r_k \ell n Q_i ,$$

for a decreasing sequence of values $\{r_k\}$ which tend to zero. The unconstrained minimizers tend to the constrained solution. The fastest method for finding the unconstrained minimizer is the generalized Newton's method which takes the form

$$x_{\ell+1} = x_\ell - \nabla^2_{xx} P_k \left(x_\ell\right)^{-1} \nabla P_k (x_\ell) t_\ell , \tag{2}$$

(where t_ℓ is the step-size scalar). The traditional way of solving the above equations is to form the $n \times n$ Hessian matrix and use Cholesky decomposition to find the desired matrix equation solution. The storage required is n^2 memory locations and the order of operations is $n^3/3$. Thus for any reasonable number of inventory items (recall $n = 2N$) this approach is computationally prohibitive. Like all large problems, there is a special structure and matrix inversion techniques can be modified to facilitate the solution of the problem.

The natural derivation of the Hessian matrix is depicted in Figure 1. If it were formed explicitly it would require $4N^2$ memory locations. Kept in its implicit form it requires only 10N locations (actually most of the numbers are already available and need not be stored separately). If N were 1,000 this is the difference between 4,000,000 and 10,000.

Computational efficiency is obtained by inverting $\nabla^2_{xx} P_k$ using the Woodbury-Morrison-Sherman formula recursively, i.e., the inverse of a matrix perturbed by an outerproduct matrix (or dyad) is the inverse of the original matrix perturbed by a dyad. Specifically,

$$\left[A + vcv^T\right]^{-1} = A^{-1} - Av\left[c^{-1} + v^T A^{-1} v\right]^{-1} v^T A^{-1} , \tag{3}$$

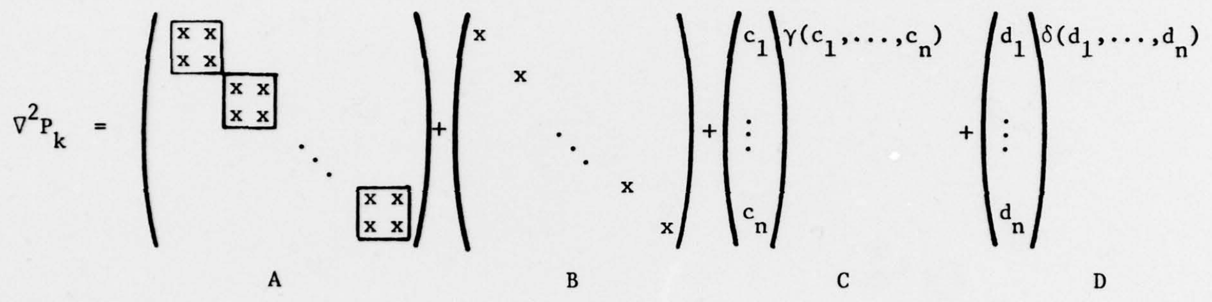


Figure 1.--Form of Hessian matrix of penalty function for inventory problem.

where A is $n \times n$, c is a scalar, and v is an $n \times 1$ vector. Applying this to (2), first $A+B$ is formed. The inverse of $(A+B)$ is gotten by inverting the 2×2 blocks. This computation requires $2N$ multiplications. The storage of this can be done in $4N^2$ (if symmetry is not used) locations. Next, two applications of the formula are given. The computation of $(A+B)^{-1}c$ requires $4N$ multiplications and the result can be stored in $2N+1$ locations. The second application of (3) yields the desired matrix inverse in implicit form. The total computations required are about $16N$ multiplications and the inverse can be stored in $7N$ memory locations. This is to be compared with $8N^3/3$ multiplications required for the Cholesky method, and $4N^2$ memory locations. Further details of this are in [McCormick 1972a].

The point is that most large nonlinear programming problems have a special structure, and algorithms for solving them can modify their matrix techniques to reduce the number of operations and computer storage. For any particular problem, the required modifications can be figured out. The question is, is there a *general* structure which occurs in all problems for which modifications to handle larger problems can be made. This point is taken up in Section 4.

Direct methods for solving nonlinear optimization problems do not handle problems of very large size. For a survey of software available and the size of problems handled see [Sandgren 1977], [Waren and Lasdon 1977], and [Wright 1978].

4. A General Approach to Large Nonlinear Programming Problems

The Lagrangian function associated with Problem (1) is

$$L(x,u,w) = f(x) - \sum u_i \, g_i(x) + \sum w_j \, h_j(x) .$$

Solving the problem can be thought of as trying to find the Karush-Kuhn-Tucker multipliers $\{\bar{u}_i\}$, and the Lagrange multipliers $\{\bar{w}_j\}$ associated with a point $\bar{x}$ solving the equations

$$\begin{aligned} \nabla_x L(x,u,w) &= 0 , & & \\ u_i g_i(x) &= 0 , & \text{for } & i=1,\dots,m \\ h_j(x) &= 0 , & & j=1,\dots,p . \end{aligned}$$

A modified form of Newton's method for solving these equations requires the inverse of the matrix

$$\begin{pmatrix} \nabla^2_{xx} L(x_k, u^k, w^k) & N(x^k) \\ N(x_k)^T & 0 \end{pmatrix} ,$$

where $N(x_k)$ is the matrix of gradients of the active constraints and equality constraints at x_k. The full matrix inverse need not be obtained, just S_k, a matrix generating the null space of $N(x_k)$, and the inverse (explicitly or implicitly of the projected Hessian), $H_k = S_k^T \nabla^2_{xx} L(x_k, u^k, w^k) S_k$. The ability of nonlinear programming algorithms to solve large problems is tied up with the ability to compute efficiently these quantities.

For large, sparse nonlinear programming problems, the techniques for computing $S(x_k)$ are in most part available from the work done in the past on linear programming problems. There is currently much work in alternative ways of computing and updating the matrix giving the null space for linear equations. Effort needs to be expended in computing H_k^{-1}. As in the inventory problem, there is no need to form H_k explicitly. From the theory of factorable programming it turns out that $\nabla^2_{xx} L(x_k, u^k, w^k)$ is given automatically as the sum of a diagonal matrix and outer product matrices. One way of utilizing this to obtain the inverse was shown in the inventory problem. There are many other linear algebra approaches for solving this problem, which can take advantage of the sparseness of the vectors defining the outer product form and the diagonal matrix. For more details see [McCormick 1978].

REFERENCES

BALINSKI, M. L. and E. HELLERMAN (eds.)(1975). *Mathematical Programming Study 4: Computational Practice in Mathematical Programming.* North-Holland Publishing Company, Amsterdam.

BEALE, E. M. L. (1975). The current algorithmic scope of mathematical programming systems, in *Mathematical Programming Study 4: Computational Practice in Mathematical Programming*, 1-11. North-Holland Publishing Company, Amsterdam.

BRACKEN, J. and G. P. McCORMICK (1968). *Selected Applications of Nonlinear Programming.* John Wiley and Sons, Inc., New York.

COHN, M. Z. and G. MAIER (eds.)(1978). *Engineering Plasticity by Mathematical Programming, Proceedings of the NATO-ASI,* University of Waterloo, August 2-12, 1977. Pergamon Press, New York (to appear).

GRIFFITH, R. E. and R. A. STEWART (1961). A nonlinear programming technique for the optimization of continuous processing systems, *Management Sci.* **7** 465-491.

HAUG, E. J., Jr. (1973). Engineering design handbook: computer aided design of mechanical systems, HMCP 706-192, U.S. Army Armament Command, Research and Engineering Directorate, Rock Island, Illinois 61201.

LASDON, L. S. and A. D. WAREN (1975). GRG user's guide, CIS-75-02, Department of Computer and Information Science, Cleveland State University.

McCORMICK, G. P. (1972a). Computational aspects of nonlinear programming solutions to large scale inventory problems, Technical Memorandum Serial TM-63488, Program in Logistics, The George Washington University.

McCORMICK, G. P. (1972b). Converting general nonlinear programming problems to separable nonlinear programming problems, Technical Paper Serial T-267, Program in Logistics, The George Washington University.

McCORMICK, G. P. (1978). Future directions: mathematical programming, in (M. Z. Cohn and G. Maier, eds.) *Engineering Plasticity by Mathematical Programming, Proceedings of the NATO-ASI*, University of Waterloo, August 2-12, 1977. Pergamon Press, New York (to appear).

MURTAGH, B. A. and M. A. SAUNDERS (1978). Large-scale linearly constrained optimization, *Math. Prog.* **14** (1) 41-72.

POWELL, M. J. D. (1978). Algorithms for nonlinear constraints that use Lagrangian functions, *Math. Prog.* **14** (2) 224-248.

SANDGREN, E. (1977). The utility of nonlinear programming algorithms, parts one and two, in *The Modern Design Series, V*, Mechanical Engineering Design Group, Mechanical Engineering Building, Purdue University, West Lafayette, Indiana 47907.

SCHRADY, D. A. and U. C. CHOE (1971). Models for multi-item continuous review inventory policies subject to constraints, *Naval Res. Logist. Quart.* **18** 451-463.

TOMLIN, J. A. (1978). Piecewise linear and polynomial approximation for dynamic programming and games, TM # 5613, Institute for Advanced Computation, Sunnyvale, California.

WAREN, A. D. and L. S. LASDON (1977). A survey of nonlinear programming software, report from Case Western Reserve University.

WRIGHT, M. H. (1978). A survey of available software for nonlinearly constrained optimization, Technical Report SOL 78-4, Department of Operations Research, Stanford University, Stanford, California 94305.